Text to Emoji Generation Using BERT😔😳😩

By Siwoo Park and Hanhee Yang

Goals and discussion

Essential goals

1. **We want to be able to import and successfully run the transformer model by fine-tuning the neural model and be able to output any emojis.**

By using a BERT-based model, we were able to successfully output appropriate emotions and map it to a set of emojis. We did this by following chronological steps of importing data, preprocessing it, defining the model, training the model, testing a piece of text, and using our final emotion vector to map to an emoji using euclidean distance and cosine similarity.

It was interesting to see that after our model was successfully trained, any text to emoji generation process could be done instantly since the model is already trained and then fine-tuned. The only thing between different texts we needed to change was the text itself, and the model would work almost instantly.

2. **We want to preprocess our dataset into a format that our transformer model can understand and get trained on.**

The text portion of the dataset was preprocessed using AutoTokenizer for the BERT model and TfidfVectorizer for the logistic regression model. The emotions were encoded as vectors using one-hot encoding.

An interesting observation was that the AutoTokenizer for BERT generated three distinct vectors for the text preprocessing (input_ids, token_type_ids, attention_mask), whereas TFidfVectorizer produced one matrix.

Example of the keys for the encoded dataset that was preprocessed.

- dict_keys(['input_ids', 'token_type_ids', 'attention_mask', 'labels'])

Example of the input ids:

- [0, 17, 48, 771, 17649, 16, 10, 159, 3207, 15, 10, 936, 47, 189, 393, 33, 2652, 1437, 50141, 37252, 1755, 11392, 4, 1437, 849, 25331, 38591, 849, 23240, 4128, 849, 605, 17649, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Here is an example attention mask:

- [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Once the input IDs are decoded, here are the results.

- ‘[CLS] “ worry is a down payment on a problem you may never have '. joyce meyer. # motivation # worry [SEP] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PA [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD [PAD] [PAD] [PAD] '

Here are examples of the labels.

- [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

These labels correspond to these emotions:

- ['anticipation', 'optimism', 'trust']

Desired goals

1. **One goal is to be able to fine tune and train the transformer model with our dataset to create a model that can predict correct emojis that align with actual labels.**

We completed this goal by testing 10 different combinations of hyperparameters (learning rate, batch size, number of training epochs, and weight decay). We found that while the accuracy did not come out to be very high 29%, this was the highest accuracy that we could achieve with our current model.

We found this section difficult because each training portion of our model would take excess of 20 minutes to complete. We also think that our model could have had a low accuracy because possibly the content of tweets don't have too much of a correlation with the emotion of the tweets.

Here are four instances of different hyperparameter combinations:

Hyperparameters	Test F1	Test ROC AUC	Test Accuracy
Weight decay = 0.1 Batch size = 8 Epochs = 5 Learning rate = 2e-5	0.7005	0.7975	0.2798
Weight decay = 0.3 Batch size = 8 Epochs = 5 Learning rate = 2e-5	**0.7178**	**0.8088**	**0.2961**
Weight decay = 0.1 **Batch size = 16** Epochs = 5 Learning rate = 2e-5	0.7055	0.7987	0.2946
Weight decay = 0.1 Batch size = 16 Epochs = 5 **Learning rate = 1e-5**	0.7073	0.8036	0.2844

2. **To be significantly better than baseline accuracy.**

The BERT model performed better in all classification accuracy metrics compared to the logistic regression model, which we used as the baseline. The F1 score was significantly better which is a strong indicator of the effectiveness of the classification.

Model	Test F1	Test ROC AUC	Test Accuracy
Logistic Regression	0.4744	0.6567	0.1681
BERT	**0.7178**	**0.8088**	**0.2961**

3. **Our goal is to develop a model that can map from text to emotion labels and then use the embeddings of emojis to map from emotion labels to corresponding emojis**.

Using the EmoTag dataset, we mapped the emotion vectors to emoji vectors by calculating the distance between the two vectors. We used cosine similarity and euclidean distance as the distance metric to compare the vectors.

The most difficult part about mapping the two vectors was formatting the output of the BERT model to match the format of the emoji vector from an entirely different dataset, but we were able to clean up the columns to keep only the essential information.

Stretch goals

1. **Emotion-specific emojis: Instead of simply mapping from predicted emotions to existing emojis, we want to be able to generate possibly new, emotion-specific emojis. This would involve training a generative model such as Variational Autoencoders (VAE) or Generative Adversarial Networks (GANs) on a large dataset of emojis, then using it to generate new emojis to correspond to specific emotions.**

Our reason for abandoning the goal of generating emotion-specific emojis in the BERT model project is the lack of a suitable dataset for training generative models like Variational Autoencoders (VAE) or Generative Adversarial Networks (GANs). Even with a suitable emotion to emoji mapping dataset, training generative models on emojis may not yield satisfactory results due to the complexity of generating visually coherent, aesthetically pleasing emojis that effectively convey specific emotions.

2. **Multi-modal input: Currently, our model is only based on text input. However, other modalities such as images or audio could be used as input to improve the accuracy of the emoji detection. An idea could be to train a separate model to detect emotions in images or audio and then combine those results with the text-based model to generate a more robust prediction.**

The reason for abandoning the goal of incorporating multi-modal input, such as images or audio, to improve emoji detection in our text-based model is the significant challenges in using computational power and time, along with the challenge of acquiring suitable datasets that are labeled with emotions. The processing and training of visual and audio data would demand substantial computational power and time, which may exceed the resources available to us.

3. **Different models: Our model is based on regular BERT. However, using slightly modified models such as RoBERTa, DistilBERT, or ALBERTA can be used to create**

models that could be slightly more accurate, faster, or improve fits to different corpora.

We successfully trained our model 3 models: BERT, RoBERTa, and DistilBERT with the same hyperparameters. Here are our results.

Text: ***'I am so shocked that you came home!'***

Model	Test Loss	Test F1	Test ROC AUC	Test Accuracy	Top Emotions for Text	Top 3 Emojis (Cosine Similarity)	Top 3 Emojis (Euclidean Distance)
BERT	0.3137	0.7005	0.7975	0.2798	Surprise	'❗', '😳', '😱'	'❗', '😳', '😱'
RoBERTa	**0.2923**	**0.7178**	**0.8088**	**0.2961**	Sadness, surprise	'❗', '😐', '🔴'	'❗', '😳', '!!'
DistilBERT	0.3109	0.6959	0.7917	0.2845	Fear, sadness	'😩','😟','😨'	'😩','😟','😨'

Metrics for BERT, RoBERTa, and DistilBERT

We found it interesting that the test accuracy in RoBERTa was higher than both BERT and DistilBERT. An intriguing observation we made in this section was that despite using the same example text, each model predicted slightly varying combinations of emotions and corresponding emojis, all of which could be justified based on the text.

Miscellaneous Goals

1. **Selecting the best distance metric to use**

One of our goals was to select the distance metric that generated the most accurate emoji based on the text. In total, we tested 4 different distance metrics (Cosine Similarity, Euclidian Distance, Jaccard Distance, Hamming Distance) but ended up only using two (Cosine Similarity, Euclidian Distance), and while there was no firm metric on how well the emojis generated were supposed to be the "correct" emoji, the Jaccard and Hamming distance generated emojis were not representative at all of the text at hand.

It was interesting to see how Jaccard distance was not able to capture nuanced relationships between intensity levels and how Hamming distance were not able to adequately capture the semantic and contextual differences between words and emotions due to the nature of how the distance metrics worked.

Text: ***'I am so shocked that you came home!'***

Cosine	Euclidean	Jaccard	Hammond
‘❗’,’😳’, ‘😱’	‘❗’, ‘😳’, ‘😱’	‘🌈’,’🌙’, ‘🌚’	‘🌈’,’🌙’, ‘🌚’

EMOJI based mappings from Emotion vectors produced from BERT model

2. **Preprocessing for logistic regression**

The preprocessing for logistic regression looked slightly different than that of BERT because we did not use the AutoTokenizer but instead the TfidfVectorizer, which assigns numerical values to words according to their frequency and importance within and across documents. Since our classification task involved understanding the importance of specific words or identifying important features, we thought that TfidfVectorizer would perform better than other methods like word embeddings.

Example of tfid vectorization

```
(0, 7949)   0.27591095322480425
```

It was interesting to see how the sentence was expanded into long vectors to capture interesting features about the sentence that might be overlooked when simply reading text.

Code and documentation

---

logistic_regression.ipynb

- This file performs multi-label text classification using logistic regression. We used TfidfVectorizer to preprocess the tweets, encode emotions using one-hot encoding, train the MultiOutputClassifier model using logistic regression, and test the accuracy of the model using accuracy, precision, recall, and f1-score.

bert.ipynb

- This file preprocesses the text data using AutoTokenizer, runs a pre-trained BERT model with BCE as the loss function, calculates accuracy metrics over a certain number of epochs, and maps the emotion vector to emoji vectors.

roberta.ipynb, distilbert.ipynb

- Similar to bert.ipynb but uses RoBERTa and DistilBERT pre-trained base and tokenizer respectively.

Reflections

What was interesting?

In our project, we explored the task of mapping emotions to emojis. We found it fascinating that output emotion vectors from transformer based models can be represented by emojis in a quantitative and mathematical way. This allowed us to effectively communicate and express seemingly boring emotions in a concise and visually appealing method which is relatable to the general public. We also discovered through this process that emojis are not always a one-to-one mapping with emotions. They are more complex and possibly subjective which makes it harder to simply predict emotions.

Additionally, we delved into the inner workings of BERT, a powerful language model, which expanded our understanding of attention mechanisms, masked language modeling, and large language models being fine-tuned on downstream tasks. By studying transformers in lectures and readings, we gained insights into the preprocessing steps necessary for our project, such as tokenizing text into input ids and incorporating self-attention mechanisms. We did not get to learn about or work on the fine tuning steps of pre-trained transformer models like BERT, so it was fascinating to see the potential of transformers in a specific task like emotion prediction.

What was difficult?

One of the difficult parts of the project was ensuring the proper formatting of the labels for multi-label text classification. It required reshaping the labels into a matrix of the shape (batch_size, num_labels) and representing them as floats to work effectively with PyTorch's BCEWithLogitsLoss. This process can be error-prone and time-consuming, especially when dealing with a large number of labels or complex label structures.

If someone were to embark on a similar project from scratch, I would strongly encourage them to consider the following recommendations to potentially save time and avoid frustrations:

1. To plan label formatting early. Devote ample time to understand the specific requirements of your multi-label classification task and plan the label formatting accordingly. Ensure you have a clear understanding of the expected shape of the label matrix and any necessary data type conversions.
2. To consider pre-processing tools. Explore pre-processing tools specifically designed for multi-label text classification tasks. These tools can simplify the process of formatting

labels and handling complex label structures. For example, libraries like scikit-multilearn provide functionalities for label transformation and manipulation.

What's left to do?

If you were going to spend another month working full-time on this project, what would you try to accomplish? Why? If Northwestern gave your group 1,000,000 USD to use for data collection or compute resources, what would they spend it on? Why?

Firstly, we could enhance the emotion classification model by fine-tuning BERT on a larger and more diverse labeled dataset. Currently, there are no large, organized datasets that have real world texts with an emoji label. One way we could approach this is by hiring assistants and crowdsource workers to collect and develop our own dataset of text and emojis by possibly sampling from text messages or Instagram captions to get a better sense of how emojis are used alongside textual context. By utilizing a dataset that contains text as data and emoji as labels to train our model, we could improve the accuracy and generalizability of the emotion prediction. Our current project assumes that the mapping of emotions to emojis is accurate, but we could quantitatively evaluate the results using this new dataset.

Additionally, we would invest in high-performance compute resources, specifically the Nvidia A100 GPU on Google Cloud, to accelerate the training and inference processes. These resources would enable us to handle larger datasets and iterate faster, ultimately improving the performance and efficiency of our emotion prediction and emoji mapping system.

Our potential long-term goal is to acquire the necessary resources to gain recognition and present our project to companies like Apple and Google, aiming to enhance their emoji recommendation system for text input. We have observed that the current recommendations provided by Apple's keyboard are limited, considering only a few words before suggesting an emoji. To address this limitation, we propose training models like DistilBERT on our curated datasets. By doing so, we can offer users not only high-quality emoji suggestions but also faster recommendations. Our aim is to collaborate with big tech companies to improve the overall user experience and provide more accurate and contextually relevant emoji recommendations within their keyboard system.